Hatching by Example: a Statistical Approach

Pierre-Marc Jodoin Emric Epstein Martin Granger-Piché Victor Ostromoukhov

Université de Montréal *

Abstract

We present a new approach to synthetic (computer-aided) drawing with patches of strokes. Grouped strokes convey the local intensity level that is desired in drawing. The key point of our approach is learning by example: the system does not know *a priori* the distribution of the strokes. Instead, by analyzing a sample (training) patch of strokes, our system is able to synthesize freely an arbitrary sequence of strokes that "looks like" the given sample. Strokes are considered as parametrical curves represented by a vector of random variables following a Markovian distribution. Our method is based on Shannon's *N-gram* approach and is a direct extension of Efros's texture synthesis models [EL99; EF01]. Nevertheless, one major difference between our method and traditional texture synthesis is the use of such curves as a basic element instead of pixels. We define a statistical metric for comparison between different patches containing various layouts of strokes. We hope that our method performs a first step towards capturing a very difficult notion of style in drawing – hatching style in our case. We illustrate our method by varied examples, ranging from typical hatching in traditional drawing to highly heterogeneous sets of strokes.

1 Introduction

Non-photorealistic rendering (NPR) is a domain of computer graphics that has grown very rapidly during the last five years. In the present contribution, we shall focus our attention on one particular trend in NPR that simulates traditional artistic media and rendering techniques. More specifically, we will explore an aspect of freehand drawing: the relation between strokes in various techniques of hatching that convey particular tone values and that obey a freely-defined notion of hatching style. Some commercial program for computer-aided illustration (e.g. Illustrator or FreeHand) offer rudimentary tools for generation of fields of hatching strokes using user-defined global parameters, like mean density, length and randomness of strokes etc. Our system for hatching by example offers a further step in sophistication and flexibility of hatching in computer-aided hatching for drawing.

Our ultimate long-term goal can be formulated as follows: we would like to be able to generate combinations of strokes that "look like" a set of strokes in a given sample patch that contains a "training set" of strokes. The sample patch can originate either from existing artwork, or can be interactively created by a user. The generated patch conveys, with more or less fidelity, the tone value desired in a synthetic drawing. The synthetic drawing system can manipulate two- or three-dimensional objects. In the case of a three-dimensional drawing system, the strokes can be seen as two-dimensional object attached to three-dimensional surface. Figure 1 illustrates this concept.

*http://www.iro.umontreal.ca/~ostrom

1.1 Previous Work

Several attempts have been carried out in the past in order to provide computer graphics support for drawing. Let us enumerate some of them that are directly related to the present work.

Winkenbach *et al.* and Salisbury *VT al.* in their pioneering work [WS96; WS94; SABS94; SALS96; SWHD97] introduced a comprehensive system for computer-aided pen-and-ink illustrations. In the most advanced version, sets of pen strokes are defined according to a user-defined vector field of orientations combined with random variations. Elber [Elb95; Elb99] and Hertzmann *et al.* [HZ00] explored geometrical properties of surfaces in order to determine automatically the direction and the spacing of the strokes in stroke-based illustrations. Sousa and Buchanan [SB99] proposed a system for simulation of the process of interaction between pencil and paper in drawing. The system produces visually plausible results; it is mainly focused on the rendering quality of the drawing process. This system does not support the higher level "style of drawing".

Durand *et al.* [DOM+01], introduced an interactive system for digital drawing with control of tonal fidelity. In their system, the user has to place each individual stroke manually. A rudimentary system for semi-automatic grouped stroke placement was proposed, but is clearly insufficient. The technique introduced in the present contribution may considerably improve the system proposed in [DOM+01]. Praun *et al.* [EPF01] proposed a real-time system that enables hatching 3D objects using a set of user-defined hatching rules producing an appearance very close to drawing. Hamel and Strothotte [HS99] introduced the *NP-templates* method used to transfer a drawing style from a 3D model to another one. Although theoretically such a system allows defining hatching rules of arbitrary complexity, in practice it is very hard to define a general style of hatching by example. Example based line art has also been addressed by Freeman *et al.* [FTP99] and Chen *et al.* [CXS+01] but none considered the strokes as being component of a MRF. We hope that our present contribution will help to open a way towards simpler and more intuitive ways to define the style of hatching in drawing.

Many other important references related to artistic rendering with strokes can be found in [GG01a] as well as on well-known web sites maintained by Craig Reynolds [Rey01] or by Amy and Bruce Gooch [GG01b] and in [DC90; Hae90; SPR+94; RK00; TF00].

We built our research on recent work done in the field of texture synthesis [WL00; WL01; Ash01] and particularly on Efros's contributions [EL99; EF01], as it will be explain in the next sections.

1.2 Problem Statement

As formulated before, our long-term goal is almost intractable because it involves many difficult problems. First, reliable detection of individual strokes in a freehand hatching image is a very difficult

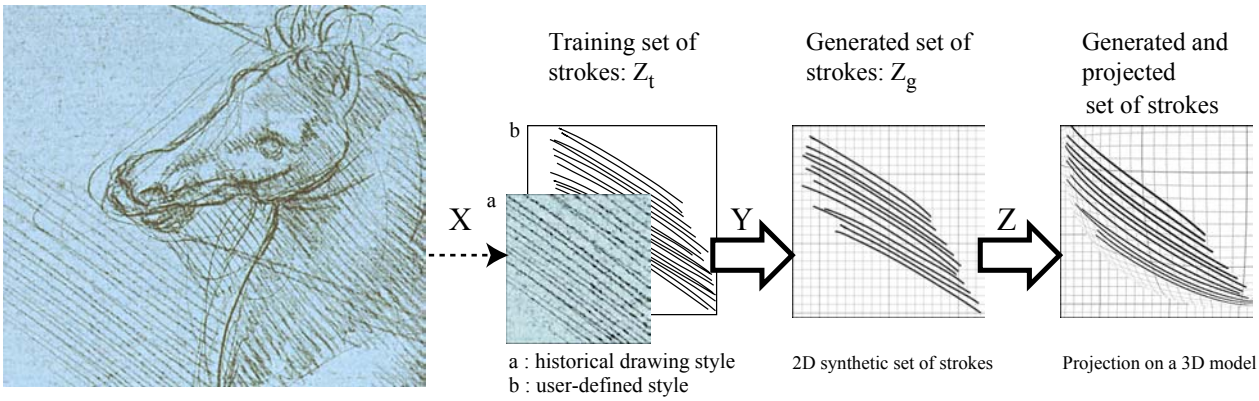

Figure 1: Our long-term goal can be formulated this way: from a given training artwork patch Z_t (or from a user-defined set of strokes gathered from an artwork, step X), generate a 2D synthetic stroke patch Z_g that "looks like" Z_t (step Y) . The generated strokes could then be used in a 2D and/or a 3D scene (step Z). In the present contribution, we shall focus only on step Y.

computer vision problem when the information about the sequencing of individual strokes is not available. Second, the criterion of similarity between the training set of strokes and the synthetic one is highly subjective by its very nature. Although some mechanisms of low vision are well understood by experimental psychologists, the interpretation of higher levels of perception – namely, the mechanisms of cognition – are far more difficult to grasp and to manipulate.

For these reasons, we will consider a simplified version of the problem statement. We shall consider only an interactive version of a one-dimensional training set of strokes that naturally imposes the sequence of the strokes through the order in which individual strokes in the training set are drawn. Let us reformulate the problem statement for the task that will be explored in the present paper (see Figure 2):

1. Given a training set Z_t, generate a new set of strokes Z_g that "looks like" it.

2. With the help of an objective visual cost function, evaluate the "visual distance" between the sets. This cost function can be statistically approximated by a `statistical_distance(`Z_t`,`Z_g`)`, as presented in Figure 3.

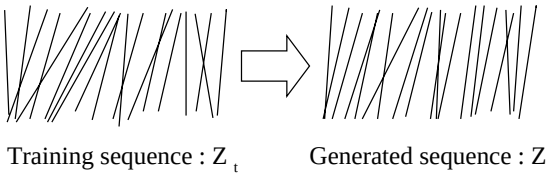

Figure 2: The objective of this paper can be formulated this way: given a training set of strokes, we want to generate a second stroke set that "looks close to it".

The way we intend to generate the stroke set Z_g is by implementing a method inspired of Efros's 1999 paper on texture synthesis [EL99]. This approach is based on Shannon's *N-gram* and thus is suitable for the problem of one-dimensional stroke sequence synthesis. The visual cost function will be a complementary tool to the stroke synthesis algorithm since it will provide an indication of the overall quality of the generated stroke set Z_g. Furthermore, because that metric will be used in parallel of the synthesis process, it will be able to indicate if and where the generated sequence diverges.

Even if many visual metrics are currently known and widely used, they are not well suited to fit our specific needs. Instead, we will characterize the pen stroke sets Z_t and Z_g via their density function $P_t(x)$ and $P_g(x)$ and find a *statistical* metric based on these values. Despite its limitations, this metric provides a tractable framework for the first-approximation evaluation of differences between sets of strokes. In order to validate our approach, we will illustrate how that statistical metric fits human perceptual quality appreciation (see Figure 3). We shall discuss the advantages and the limitations of our approach in the appropriate sections.

1.3 Model Description

Let us consider a stroke sequence Z_t similar to the one presented on Figure 2. This sequence can be formally represented by the set $Z_t = \{x_0, x_1, x_2, \ldots, x_n\} = x_0^n$ where x_i is a stroke defined by a vector of parameters associated with a random variable X_i. We consider Z_t to be a realization of a Markov Random Field (MRF). In other words, we consider that the conditional probability of a random variable X_i associated with a stroke depends on its close neighborhood:

$$p_t(X_i = x_i | x_0^i) = p_t(X_i = x_i | x_{i-N}^{i-1}) \tag{1}$$

where x_0^{i-1} is the set of all strokes appearing before x_i and x_{i-N}^{i-1} is the sequence of $N-1$ strokes located before x_i. Equation (1) is a Markovian model of order $N-1$ [Ben99]. The stroke set Z_g that we want to generate needs to be different from Z_t while having the same conditional probability:

$$p_t(X_i = x_i | x_{i-N}^{i-1}) = p_g(Y_i = y_i | y_{i-N}^{i-1}) \tag{2}$$

where Y_i is the random variable associated with the generated strokes $y_i \in Z_g$. Because of this, we made the decision to base our model on Shannon's *N-gram* [Sha51] which is a simple markovian approach that was recently use in the context of texture synthesis [EL99; EF01; WL00; WL01]. The main advantage of Shannon's method is its algorithmical simplicity combined with a fairly good overall visual quality of the results. Nevertheless, the *N-gram* method is not perfect and, as Efros has mentioned [EL99], it has a tendency to diverge after a few iterations. When it diverges, the difference between the generated set Z_g and the training set Z_t is clearly visible. For this reason, we need to implement an objective metric that will indicate how visually different the generated stroke set is from the training set. This tool will also help us locate where the method starts to diverge. We made the assumption that if Shannon's method is an appropriate one for the problem we are trying

to solve, the generated set's conditional distribution $P_t(X_n|X_{n-N}^{n-1})$ should be statistically "close" to the training set's conditional distribution $P_g(Y_n|Y_{n-N}^{n-1})$. The distance between these two distributions will give us an indication of how well suited our method is. In other words, we intend to make a relation between this statistical metric and the human visual appreciation of the difference between the stroke sequences.

1.4 Our Contribution

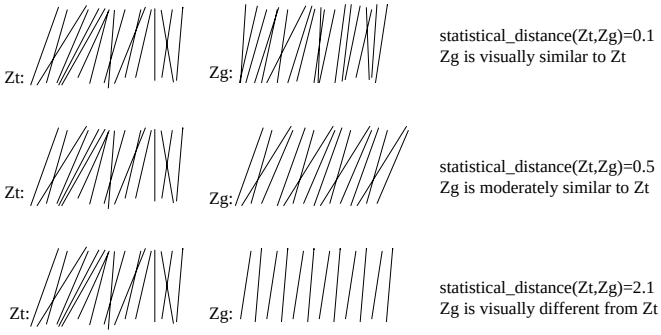

Figure 3: This illustrates the relationship between the statistical metric and the visual impression.

This paper has two main contributions:

1. Propose an *N-gram* based method to synthesize a sequence of strokes Z_g that "look like" a given training set Z_t. By doing this, we are approaching the notion of style of hatching in drawing.

2. Find a statistical metric that correlates the human visual perception in the context of stroke synthesis. This metric will be used to evaluate the overall quality of Z_g compared to Z_t while detecting if and when the method diverges.

The remainder of this paper is organized as follows. In section 2, we will formalize the problem by elaborating the statistical basis of our approach. Section 3 shows some results while section 4 makes reference to the method limitations. Section 5 discusses conclusion and future work.

2 Underlying Statistical Theory

2.1 The Generation Model

Let us consider $Z = \{x_0, x_1, x_2, ..., x_{k-1}\} = x_0^{k-1}$ a given sequence of strokes where x_i is a stroke and its associated random variable X_i follows a conditional probability $p(X_i|X_{i-N}^{i-1})$. Each stroke x_i is defined as a parametrical curve and is represented by a vector of dimension d where d depends on the complexity of the curve. Because of the specific needs of our algorithm, all strokes in Z has to have the same dimension d. As shown in Figure 4, the objective is to find a stroke x_k that would best fit at the end of the sequence.

In the 1940's, Shannon studied the problem of language prediction which is, in many respects, similar to the stroke synthesis problem. According to a statistical knowledge of the English language, his objective was to predict the next letter of a text when the previous $N-1$ ones are known. He came up with a solution by introducing the famous F_N measure called the *N-gram entropy*. *It measures the average uncertainty (conditional entropy) of the next letter when the preceding $N-1$ are known* [Sha51]. In the present

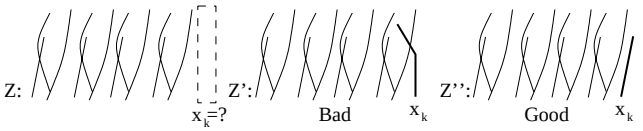

Figure 4: According to a given stroke sequence Z, the goal of our method is to find the best stroke x_k to put at the end of it. A good stroke is one that respects the preceding stroke pattern.

contribution, we replace the letters by strokes and the English sentences by stroke sets. In that context, the F_N measure is given by

$$F_N = -\sum_v p(x_{k-N}^{k-1}, x_v) \log p(x_v|x_{k-N}^{k-1}) \tag{3}$$

where x_{k-N}^{k-1} are the last $N-1$ strokes so far generated, $p(x_{k-N}^{k-1}, x_v)$ the joint probability of the N-gram sequence (x_{k-N}^{k-1}, x_v) and $p(x_v|x_{k-N}^{k-1})$ the conditional probability of having stroke x_v given the last $N-1$ strokes. $p(x_{k-N}^{k-1}, x_v)$ can be understood as being the "word frequency", which is (in English prediction), the relative frequency of a given word in the English language. For example, the most frequent English word "the" has probability 0.071 while the second most frequent word "of" has probability 0.034 [Sha51]. In the context of stroke synthesis, we do not have a dictionary of words that could inform us of the probability of having a certain sequence of strokes. For this reason, we consider all stroke sequences of size N to be equally probable. This leads us to a simpler N-gram entropy measure

$$F_N = -\sum_v \frac{1}{T_N} \log p(x_v|x_{k-N}^{k-1}) \tag{4}$$

where the constant T_N is the total number of stroke sequences having length N. Intuitively, we might want to find the *best* stroke for which $p(x_v|x_{k-N}^{k-1})$ is maximal. Doing this would be appropriate if the training set was very large. But in a context where that set is relatively small, using only the *best* stroke would turn our problem into a deterministic one. To avoid that, instead of finding only one best stroke, we shall consider S, the set of all *plausible* strokes

$$S = \{x_i | p(x_i|x_{k-N}^{k-1}) > c \text{ AND } x_i \in Z\} \tag{5}$$

where c is a constant and x_{k-N}^{k-1} are the last $N-1$ strokes so far generated. As Efros previously suggested, we linked the local conditional probability with the *Euclidean distance* between the two neighborhoods x_{k-N}^{k-1} and x_{i-N}^{i-1} via the following Gibbs probability:

$$p(x_i|x_{k-N}^{k-1}) \propto e^{-\|d(x_{k-N}^{k-1}, x_{i-N}^{i-1})\|} \tag{6}$$

where $d(.)$ is the Euclidean distance between x_{k-N}^{k-1} (the last $N-1$ strokes in the sequence) and x_{i-N}^{i-1} (the $N-1$ strokes preceding x_i). The new stroke x_k is taken randomly from S. Here is an algorithm that schematically describes how the generated set Z_g is synthesize from the training set Z_t:

Function `GENERATE_STROKES`(Z_t,S_t,S_g,N):
. Allocate Z_{temp}, a temporary array of strokes having size $S_t + S_g$
. Allocate Z_g, an array of strokes having size S_g
. $Z_{temp} \leftarrow Z_t$
. **For** i from 1 to S_g **Do**
. S $\leftarrow$ `GET_PLAUSIBLE_STROKES`(Z_t,N)
. $x_i \leftarrow$ pick randomly one stroke from S

. Append x_i to Z_{temp}
. Append x_i to Z_g
. **Return** Z_g

Function GET_PLAUSIBLE_STROKES(Z_t,N):
. Allocate S, the list of plausible strokes
. $z_l \leftarrow$ the last $N-1$ strokes of Z_t
. **For** all sequences z_c of length $N-1$ in Z_t **Do**
. $dst \leftarrow$ EUCLIDEAN_DISTANCE(z_c,z_l)
. **If** $e^{-||dst||} > c$ **AND** $z_c \neq z_l$ **Do**
. Append z_c to S
. **Return** S

2.2 Statistical Distance Between the Training and Generated Sets

Now that we have a method to generate the stroke sequence Z_g from a training sequence Z_t, we want to know how "far" these two sequences are from each other. In other words, does the sequence generated by the *N-gram* algorithm have the same statistical properties as the training one? To compute that "distance", we use the respective conditional distributions $P_t(X_k|X_{k-N}^{k-1})$ and $P_g(Y_k|Y_{k-N}^{k-1})$ and calculate their *Kullback-Leibler distance* $KL(.)$ (or *asymmetric divergence* [Bis95])

$$KL_1 \ : \ KL(P_t(x_k|x_{k-N}^{k-1})||P_g(y_k|y_{k-N}^{k-1})) = -\int_{-\infty}^{\infty} P_t(x_k|x_{k-N}^{k-1}) ln \frac{P_g(y_k|y_{k-N}^{k-1})}{P_t(x_k|x_{k-N}^{k-1})} dx \quad (7)$$

Because $P_t(.)$ is the training set distribution, we hope that $P_g(.)$ is as close as possible to it. In fact, it can be easily shown that equation (7) reaches its minimum when $P_t(.) = P_g(.)$ and gets bigger when $P_g(.)$ differs from $P_t(.)$

2.3 The Curse of Dimensionality

The conditional distributions $P_t(X_k|X_{k-N}^{k-1})$ and $P_g(Y_k|Y_{k-N}^{k-1})$ are generally not known *a priori*. To compute these functions, we have to empirically estimate them from the training strokes set Z_t and the generated set Z_g. With an *N-gram* window width N and a stroke dimension d, the density estimation process will have to deal with a total of $D = N \times d$ dimensions. A serious problem called *the curse of dimensionality* [Bis95; Bel61] arises when D reaches large values. For instance, if we divide each dimension into a discrete set of M bins, the strokes will be located in a world of M^D bins. This implies that the number of bins grows exponentially with the stroke complexity and the window width. In this way, if we have a small number of strokes dispersed into M^D bins (where D is large), most of the bins will be empty and the density estimation process will fail to accurately estimate the density function. Consequently, a reasonable estimation for $P_t(X_k|X_{k-N}^{k-1})$ and $P_g(Y_k|Y_{k-N}^{k-1})$ would require an enormous stroke database and would bring our problem towards intractable solutions. To work around that dead-end, we decided to replace the conditional distributions $P_t(X_k|X_{k-N}^{k-1})$ and $P_g(Y_k|Y_{k-N}^{k-1})$ by the *a priori* distributions $P_t(X)$ and $P_g(Y)$. The Kullback-Leibler distance then becomes

$$KL_2 \ : \ KL(P_t(x)||P_g(y)) = -\int_{-\infty}^{\infty} P_t(x) ln \frac{P_g(y)}{P_t(x)} dx \quad (8)$$

From a formal point of view, making such a simplification may appear clumsy and inappropriate since equation (8) is not an approximation of equation (7). Furthermore, knowing that KL_1 is the true conditional distance between the two stroke sequences, we may wonder how KL_2 can give any good information on KL_1.

We stated at equation (2) that the generated strokes in Z_g need to be sequenced the same way they are in Z_t. To be true, one inherent condition is to have two similar *a priori* distributions. In other words, the two sequences need to use the same strokes and at the same rate. Consequently, when KL_1 is small, it implies that KL_2 is small too:

$$KL_1 \text{ is small} \Longrightarrow KL_2 \text{ is small} \quad (9)$$

But because of the forward direction of this statement, having a small KL_2 implies nothing about KL_1's value. On the other hand, we can say that when KL_2 is large KL_1 has to be large too:

$$KL_2 \text{ is large} \Longrightarrow KL_1 \text{ is large} \quad (10)$$

In other words, when KL_2 is large, it tells us that the strokes in Z_t are generally not the same as those found in Z_g. Under these conditions, there is no way Z_g can be similar to Z_t. KL_2 is thus only meaningful when its value is large since it provides an indication of what the *true* conditional distance between the sets Z_g and Z_t is.

We mentioned earlier that our algorithm has a tendency to diverge. When testing our method, we empirically observed that the first generated strokes were generally well chosen. But when the system starts to diverge, it often loops over and over on the same stroke subset. This frequent problem is one that can be efficiently detected by the metric KL_2 because whenever KL_2 is large, it implies that KL_1 is too. In other words, when the method starts to reuse the same strokes over and over, KL_2 reaches large values telling us that KL_1 (the *true* distance) is also getting larger.

2.4 Density Estimation

The *a priori* distribution $P(x)$ of a stroke set Z is not known at first and thus needs to be estimated. We know from the theory of probability that if a random variable X has a density function $G(X)$ and a probability q(x), then

$$G(x) = \lim_{h \to 0} \frac{1}{2h} q(x-h < x < x+h) \quad (11)$$

Given a list of points and a given window h, we can compute $q(x-h<x<x+h)$ using a *naive estimator* [Sil86] by calculating the number of points x_i falling into the interval $[x-h, x+h]$ as follows:

$$\hat{G}(x) = \frac{1}{2hn} < \text{no. of } x_i \text{ falling in } [x-h, x+h] > \quad (12)$$

where n is the total number of points in the list. We can rewrite equation (12) using a *kernel function K* (known as the *Parzen window* [Bis95; Sil86])

$$\hat{G}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_{g_i}}{h}) \quad (13)$$

where h is called the *smoothing parameter* and K satisfies the condition

$$\int_{-\infty}^{\infty} K(x) dx = 1 \quad (14)$$

A common choice for that kind of kernel is the Gaussian function. This leads to

$$\hat{G}(x) = \frac{1}{nh} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} exp\{-\frac{(x - x_{p_i})^2}{2h^2}\} \quad (15)$$

Combining equation (15) to the stroke density estimation problem gives us

$$P(x) \approx \frac{1}{n}\sum_{i=1}^{n}\frac{1}{(2\pi h^2)^{d/2}} exp\{-\frac{\|x-x_i\|^2}{2h^2}\} \quad (16)$$

where $x_i \in Z$ is a stroke of dimension d. If Z contains normally distributed data with unit variance, we can compute the optimal *smoothing parameter* h_{opt} [Sil86] instead of using a constant h

$$h_{opt} = (\frac{4}{2d+1})^{\frac{1}{d+4}} \times n^{-\frac{1}{d+4}} \quad (17)$$

where n is the size of Z and d is the dimension of the strokes. The density estimation process can be summarized by the following pseudo-code:

Function `DENSITY_ESTIMATION`(Z)
. $d \leftarrow$ the dimension of the strokes in Z
. $n \leftarrow$ the number of strokes in Z
. **If** the strokes are normally distributed with unit variance **Then**
. h $\leftarrow$ `COMPUTE_OPTIMAL_H`(d,n) /* eq.(17) */
. **Else**
. $h \leftarrow$ a constant value
. $P_g(x) \leftarrow$ `ESTIMATE_DENSITY`(d,n,h,X) /* eq.(16) */
. **Return** $P_g(x)$

3 Implementation and Results

3.1 Requirements and Implementation

As we stated previously, our implementation contains two complementary parts: the *N-gram* algorithm and the statistical cost function KL_2 that comes with it. While testing our program, we found out that the stroke generation algorithm is very sensitive to the choice of the window width N, the form of the strokes and their layout. In fact, as a rule of thumb, there are two things we can say. First, the more complex the strokes are, the bigger the training sequence needs to be. Secondly, highly correlated sequences such as those presented on Figure 7 always require a large N. Fortunately, most of the time when one of these two conditions is not respected, the cost function KL_2 reaches large values and warns us that something goes wrong. In such a case, the synthesis process is stopped and the problematic curves are eliminated. The process can then be restarted over the last strokes so far generated with a new set of parameters (typically, a larger window size N).

Its important to understand that this KL_2 "large value" is not a universal and absolute threshold above which the resulting sequence is *always* visually bad. This threshold depends on the stroke's form and the sequence complexity. Typically, a simple sequence such as the one on Figure 2 will have a threshold around 0.4. On the other hand, a more complex one such as Figure 7 (a) will have a threshold located near 0.25. After all the tests we have made, we can say that for a majority of stroke sequences, the related threshold is located somewhere between 0.2 and 0.5.

This threshold concept is fuzzy because it is directly related to the subjective appreciation of the user. An acceptable result for one may not be it for another. The KL_2 threshold evaluation is thus left to the user.

Our implementation is simple and straightforward. Depending on the choice of the user, the strokes can be Bezier curves or straight lines. In either case, a stroke is always represented as a vector of variables and the stroke sequences as an array of vectors. As shown in function `GENERATE_STROKES(.)`, during the synthesis phase, the generated strokes are always picked up in the training set Z_t and copied in the generated set Z_g.

3.2 Results and Applications

In the present section, there are three major questions we intend to answer:

1. Does KL_2 correspond to the human visual perception?
2. Using the KL_2 metric, is it possible to detect when the algorithm starts to diverge?
3. How can we generalize this technique to two-dimensional drawing?

In order to answer the first two questions, we manually created several stroke sequences Z_t and launched our algorithm over them. We then closely observed the relation between the result quality of Z_g and the value of KL_2.

At first, we generated three sequences Z_g from a single training sequence Z_t. As shown in Figure 5, we kept constant the size of Z_g and Z_t but varied the window width N. We can clearly see that a window width of 4 is inappropriate regarding the visual quality. The value of KL_2 also corroborates this observation since it is nearly 10 times larger than it was for the first sequence.

We then generated long sequences and tried to detect when and where the method starts to diverge. As presented in Figure 6, we first drew a series of 41 strokes by hand. From this set, we generated 20 consecutive sequences of 50 strokes and calculated their respective KL_2 values. The resulting Z_g size is thus 1000 strokes. We realized that the 19th sequence (containing strokes 900 to 950) had a very large KL_2. This was because the algorithm started to loop over the same stroke subset. A close inspection showed that the method started to diverge around stroke 910.

We also generated highly correlated sequences where the order in which the strokes are placed is crucial. As shown in Figure 7, our method was able to successfully generate words made out of strokes as well as a cloud of small crosses.

All stroke generation were done interactively and required a small amount of memory. Typically, from a training set of 50 hatches, our method requires no more than 2 seconds and 50KB of memory to synthesize 500 strokes. We made these tests on a 1.4GHz Athlon processor.

The last question to answer is more tricky since our method was built to generate linear sequences of strokes and do not *a priori* address the problem of 2D drawing. Nevertheless, we overcame that limitation by carefully incorporating the artist in the drawing process. That drawing process is simple : when a stroke based image is to be rendered, the artist is asked to use a brush to specify where the linear sequences of strokes are to be placed in the image. This idea was already explored by [DOM+01] who showed that this approach brings successful results. The linear stroke sequences are generated on the fly by our method using a given training set of strokes Z_t. That way, the artist is not evicted from the creation process and have an intuitive way to freely distribute the strokes on the 2D image. A concrete example of that drawing process is shown in Figure 8. Such a tool manages the "mechanical" aspect of the stroke generation process and leave the artist in control of the aesthetic aspect of the artwork. More drawing examples using this method are presented in Figure 9.

4 Limitations

Our method suffers from two inherent limitations. The first one comes from the fact that our method is not a "true" stroke generation algorithm in the sense that no strokes are really generated. This *N-gram* based method picks up strokes from the training set and duplicates them in the generated set. Consequently, when Z_g is very much bigger than Z_t, it contains a large number of strokes but with

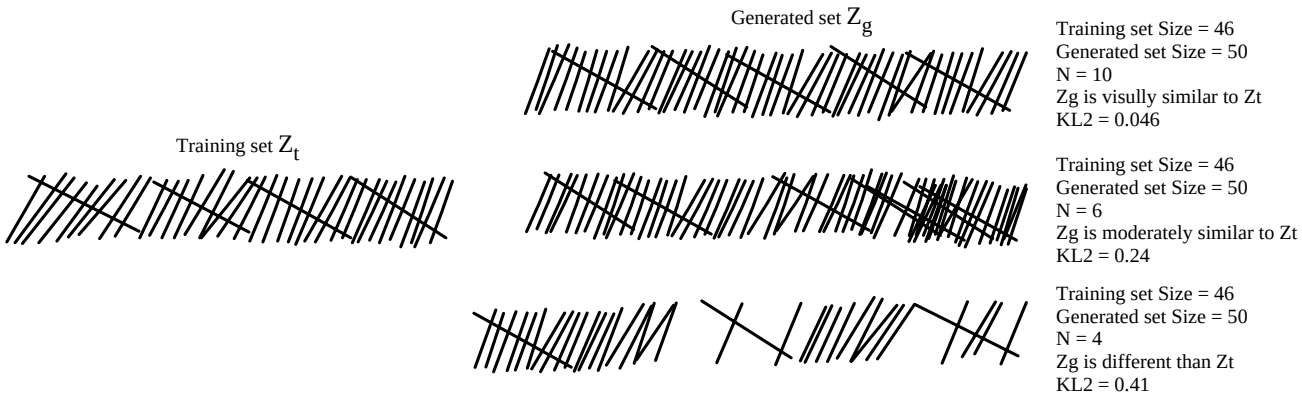

Figure 5: Starting from Z_t, we generated three different sets all having size 50. The first one was computed using a window width of 10, the second one 6 and the last one 4. It can be seen that the third sequence is visually different from Z_t and its related KL_2 is large.

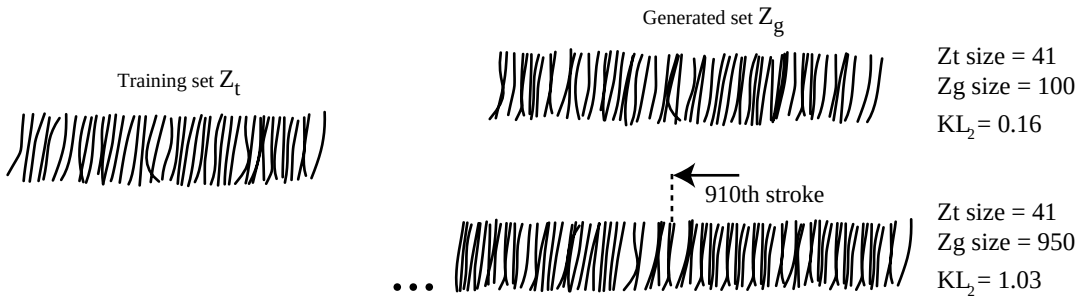

Figure 6: From a short hand-drawn sequence Z_t of size 41, a long Z_g sequence is synthesized one block of 50 strokes after the other. After generating 2 blocks, KL_2 is not large and the result is visually similar to Z_t. On the other hand, after synthesizing 950 strokes (19 blocks), KL_2 gets very large telling us something wrong is going on with the result. A closer inspection showed that the method started to generate the same stroke subset over and over around stroke 910.

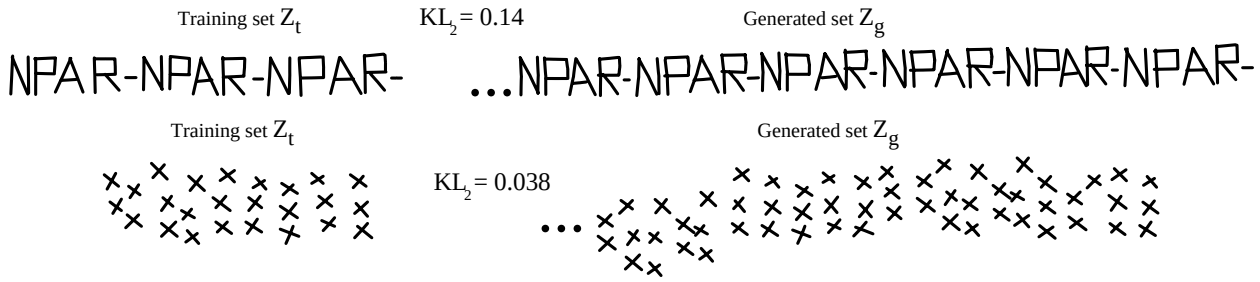

Figure 7: Highly correlated sequences generated successfully using our method.

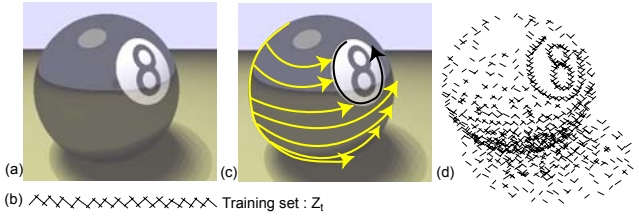

Figure 8: Image (a) is the scene the user wants to draw and (b) is the training set of strokes. (c) illustrates some drawing paths entered by the artist and (d) is the final result.

a limited variety of form. This leads to the situation where equation (2) fails to be true and KL_2 loses its meaning. In other words, the KL_2 value makes sense only when the two sequences have similar number of strokes. This limitation is not crucial in computer-aided drawing since generation of long X_g sequences can be made block by block as shown in Figure 6. That way, KL_2 is always meaningful while bringing a solution to the divergence problem.

The second limitation comes with the meaning of the statistical metric KL_2. As we said earlier, this metric represents the statistical distance between two density functions that do not take into account the conditional aspect of the stroke sequences. It pays attention only to the general distribution of strokes and not to how they are put together. Consequently, a low KL_2 value does not imply that the visual distance between the two sets is small. KL_2 is rather a *negative* indicator that underlines visually unpleasant re-

sults when it gets large. Fortunately, degenerated cases where the statistical metric KL_2 does not fit the visual perception are rare. KL_2 have been useful for the majority of examples we have tried and we believe that this approach provides a first step towards a more versatile metric for comparison between visual objects.

5 Conclusion

In this paper, we have presented a new approach to synthetic computer-aided hatching, engraving and similar artistic techniques. The goal here was to grasp the notion of style from typical hatching. We believe that the style of a stroke sequence is given by the form, the orientation and the layout of the strokes. Thus, the major difference between our method and traditional texture synthesis comes with the use of parametric strokes as a basic element instead of pixels.

From a given hand-drawn training sequence, our method attempts to generate a new sequence that has the same visual properties. To achieve this task, we came up with an *N-gram* based method close to the one presented by Efros [EL99] in 1999. Our method considers stroke sequences as being a realization of MRF and tries to generate a sequence of strokes having the same conditional probability as the training set – see equation (2). This method is simple, fast and does not require a lot of memory. On the other hand, it has an unpleasant tendency to diverge and is very sensitive to its parameters.

To overcome these inherent limitations, we looked for a visual cost function that would provide us with an objective quality measure. We came up with the idea of using a statistical cost function that would evaluate how "far" visually the generated set is from the training set. We made the choice of the *Kullback-Leibler* distance (KL_1) between the conditional distribution of the two sequences. However, we showed in section 2.3 that this measure falls unconditionally into the *Curse of dimensionality* and that for this reason, it could not be kept. We bypassed that problem by replacing the conditional distributions by the *a priori* distributions. The conditional *Kullback-Leibler* distance KL_1 became the *a priori Kullback-Leibler* distance KL_2. This last measure provided useful information on the overall quality of the results and helped us locate when and where the method diverges. KL_2 is a measure that indicates how *far* statistically the two sequences are from each other but gives no clue on the visual closeness of the sequences.

The proposed method can be used in programs for computer-aided illustration and/or drawing where the user provides interactively a training set of hatching or make reference to a library of hatchings and automatically applies this style on a selected area of a synthetic drawing. Our method can also be immediately used as a brush feature by graphic programs such as Gimp, Photoshop, Illustrator or by any other applications going beyond this spectrum.

6 Future Work

While writing this paper, we came to realize that our contribution raises more questions than it solves. For instance, we do not know what the mathematical relation between the *window width* N, the *training set size*, the *generated set size* and the stroke dimension really is. We thus intend to encompass this relation in order to be able to further stabilize the algorithm.

We also are looking for an innovative method that would generate new strokes instead of reusing the same ones over and over as is presently the case. Such a solution would fully satisfy equation (2), whatever the size of the generated stroke set.

In the actual version of our work, we do not control the tone of the strokes. We only fix it to a constant value, whatever the shape or the position of the strokes may be. We thus are currently working to link the tone of the strokes with a given input picture in order to help smooth out the shades. Varing the width of the strokes and using a method inspired of the one proposed by Praun *et al.*[EPF01] are amoung the possibilities.

As we stated in the introduction, one of our long-term goals is to be able to extract the style of a work of art in order to apply it in a 2D or 3D scene. Understanding what the drawing style is, extracting it from a sample of artwork and transferring it into a scene are still problems for future work. In this perspective, the present contribution represents only one piece of the puzzle. Nevertheless, we believe that it is a useful first step in the right direction.

Acknowledgment

The authors would like to give a special thank to Max Mignotte, Sebastien Roy, Justin Bur and Fredo Durand for their precious help. We also thank anonymous reviewers for insights and comments.

References

M. Ashikhmin. Synthesizing natural textures. *2001 ACM Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton, New Jersey, U.S.A., 1961.

Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1999.

Ch. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford, 1995.

H. Chen, Y. Xu, H.-Y. Shum, S. C. Zhu, and N. Zheng. Example-Based facial sketch generation with non-parametric sampling. In *Proc. of ICCV-2001)*, pages 433–438, Los Alamitos, CA, 2001.

D. L. Dooley and M. F. Cohen. Automatic Illustration of 3D Geometric Models: Lines. In *Proc. of Symposium on Interactive 3D Graphics*, pages 77–82, New York, 1990.

F. Durand, V. Ostromoukhov, M. Miller, F. Duranleau, and J. Dorsey. Decoupling strokes and High-Level attributes for interactive traditional drawing. *Proc. of the 12th Eurographics Workshop on Rendering*, pages 71–82, 2001.

A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *proc of SIGGRAPH 2001*, pages 341–346, 2001.

A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision (ICCV'99)*, pages 1033–1038, 1999.

G. Elber. Line art rendering via a coverage of isoparametric curves. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):231–239, 1995.

G. Elber. Interactive line art rendering of freeformsurfaces. In *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 1–12, 1999.

M. Webb E. Praun, H. Hoppe and A. Finkelstein. Real-time hatching. In *proc. of SIGGRAPH 2001*, pages 579–584, 2001.

W. T. Freeman, J. B. Tenenbaum, and E. Pasztor. An example-based approach tp style traslation for line drawings. Technical Report TR-99-11, MERL – A Mitsubishi Electric Research Laboratory, 1999.

Amy Gooch and Bruce Gooch. *Non-Photorealistic Rendering*. 2001.

Amy Gooch and Bruce Gooch. Non-photorealistic rendering. An annotated survey of online resources, http://www.cs.utah.edu/npr/, 2001.

P. E. Haeberli. Paint by numbers: Abstract image representations. *Proc. of SIGGRAPH 90*, 24(4):207–214, August 1990.

J. Hamel and T. Strothotte. Capturing and re-using rendition styles for non-photorealistic rendering. In *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 173–182, 1999.

A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *proc. of Siggraph 2000*, pages 517–526, 2000.

C. Reynolds. Stylized depiction in computer graphics non-photorealistic, painterly and 'toon rendering. An annotated survey of online resources, http://www.red3d.com/cwr/npr/, 2001.

C. Rössl and L. Kobbelt. Line-art rendering of 3d-models. *8th Pacific Conference on Computer Graphics and Applications*, pages 87–96, October 2000.

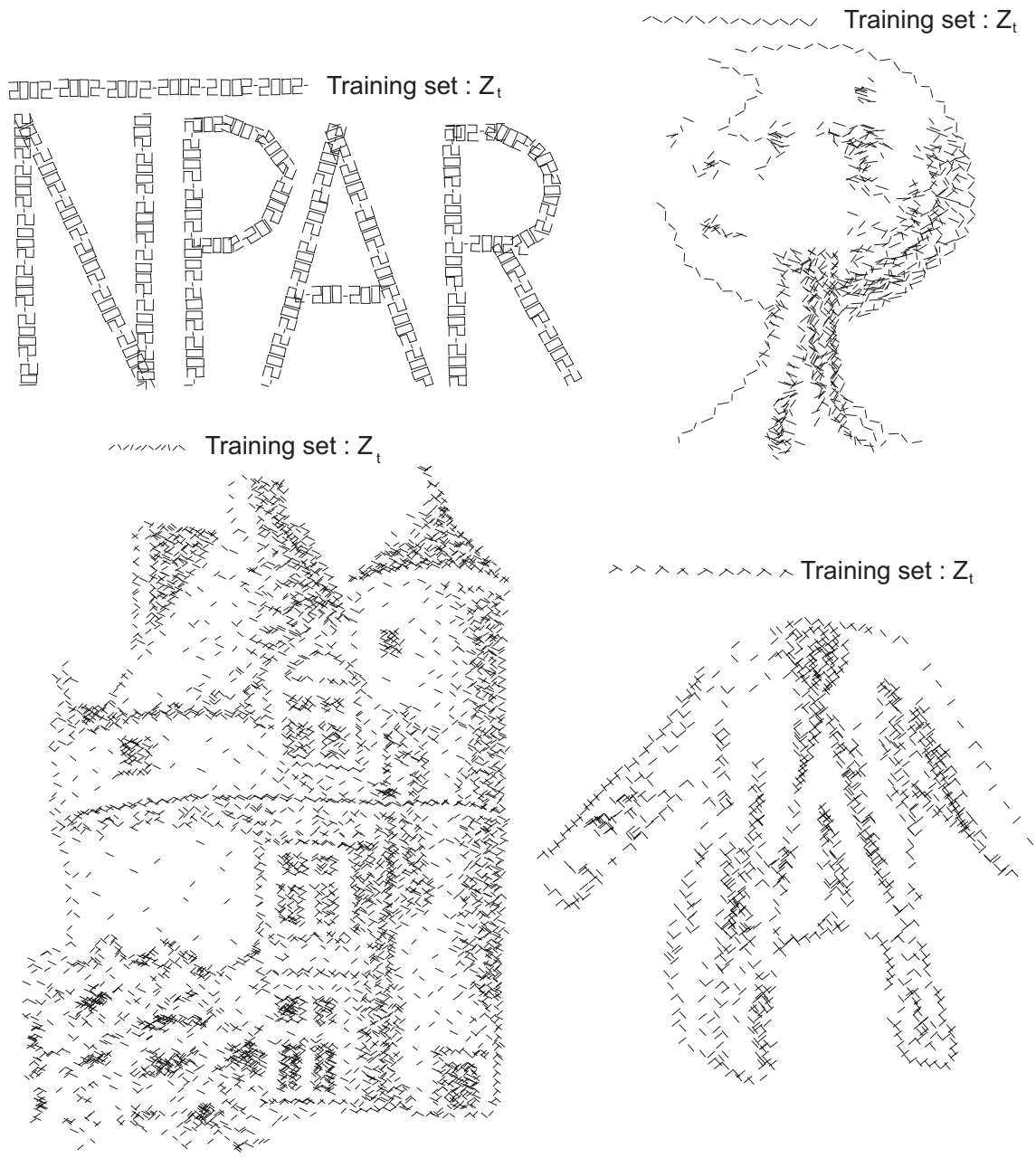

Figure 9: These for images were generated interactively with a brush that generates stroke sequences similar to the training set Z_t.

M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. *Proc. of SIGGRAPH 94*, pages 101–108, July 1994.

Mike Salisbury, Corin Anderson, Dani Lischinski, and David H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. In *proc. of SIGGRAPH 96*, pages 461–468, 1996.

M. Sousa and J. Buchanan. Computer-Generated Graphite Pencil Rendering of 3D Polygonal Models. In *Proc. of EuroGraphics'99*, pages C195–C207, 1999.

C. E. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, January 1951.

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. London, 1986.

T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey. How to render frames and influence people. *Eurographics 94*, 13(3):C/455–C/466, 1994.

M. P. Salisbury, M. T. Wong, J. F. Hughes, and H. Salesin D. Orientable textures for image-based pen-and-ink illustration. *Proc. of SIGGRAPH 97*, pages 401–406, 1997.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *proc. of Siggraph 2000*, pages 479–488, 2000.

Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH 2001*, pages 355–360, 2001.

G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. *Proc. of SIGGRAPH 94*, pages 91–100, 1994.

G. Winkenbach and D. H. Salesin. Rendering Parametric Surfaces in Pen and Ink. In *proc. of SIGGRAPH 96*, pages 469–476, 1996.